UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/897,552	07/02/2001	James A. Sievert	RA 5202 (CST 1028.1120101	9692

27516 7590 05/31/2005

UNISYS CORPORATION
MS 4773
PO BOX 64942
ST. PAUL, MN 55164-0942

EXAMINER	
ROCHE, TRENTON J	

ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 05/31/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/897,552	SIEVERT, JAMES A.	
	Examiner	Art Unit	
	Trent J. Roche	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *16 November 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-8 and 10-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-8 and 10-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *02 July 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This office action is responsive to communications filed 16 November 2004.

2. Per Applicant's request, amended claims 4, 5, 7, 8 and 10 have been entered. Claims 9 has been canceled. Newly added claims 13-24 have been entered.

3. Claims 1-8 and 10-24 have been examined.

Claim Rejections - 35 USC § 101

4. In view of the Applicant's amendments, the rejection of claims 7-9 under 35 U.S.C. § 101 has been withdrawn.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

6. Claims 1-8, 10, 13-17 and 19-23 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent 6,125,364 to Greef et al, hereafter referred to as Greef.

Per claim 1:

Greef discloses:

- a method for creating a plurality of objects from data in persistent storage ("The class identifier is used to invoke the object creation method on the correct class object..." in col. 6 lines 50-52)
- the objects having object pointers, unique object identifiers, and object types as attributes ("Each entity in the system has a unique object identifier and a class identifier...objects have lists of smart pointers..." in col. 4 lines 21-41)
- reading the total number of type sets (Note Figure 10, item 401. The offset represents the total number of type sets.)
- for each type set, read the total number of objects in each type set (Note Figure 10, item 402)
- for each object in said type set, creating an object form said data in persistent storage (Note Figure 10, item 404)
- for each object pointer in said objects, obtaining the unique object identifier corresponding to said object pointer ("The smart pointer asks the Class that corresponds to the objects's class identifier..." in col. 4 lines 59-60)
- for each obtained unique object identifier, obtaining the object address corresponding to said unique object identifier and setting each of said object pointers to said corresponding object address ("The smart pointer asks the Class that corresponds to the objects's class identifier, to create an object of that class" in col. 4 lines 59-61. It inherently obtains the address so that it can create the object.)

substantially as claimed.

Per claim 2:

The rejection of claim 1 is incorporated, and further, Greef discloses objects created in a first pass and said objects' pointers values are set in a second pass subsequent to the first pass as claimed (Note Figure 10, items 404 and 407 and the corresponding sections of the disclosure. The object is created first, then the data objects are populated.)

Per claim 3:

The rejection of claim 1 is incorporated, and further, Greef discloses attempting to obtain object addresses, and setting object pointers equal to said object address if said pointed to objects exist, otherwise deferring said setting object pointer step until object exists as claimed ("When an operation is invoked on a smart pointer, the object that is points to is faulted into the entity cache if it does not already exist. This is done by invoking an operation on the Data Store that can find an object...A smart pointer is then created and the smart pointer asks the Class that corresponds to the object's class identifier..." in col. 4 lines 52-60.)

Per claim 4:

Greef discloses:

- a method for writing a plurality of objects in non-persistent storage to persistent storage ("Figure 7 is a flowchart demonstrating 'storing' objects on a flexible nonvolatile or persistence memory" in col. 5 lines 49-50)
- the objects having pointers to objects, unique object identifiers, and object types as attributes ("Each entity in the system has a unique object identifier and a class identifier...objects have lists of smart pointers..." in col. 4 lines 21-41)

- providing one or more common interfaces that are used by each of the plurality of objects to write the objects from non-persistent storage to persistent storage ("The two primary classes for streaming objects into and out of persistence is the Data Base Cursor and the Data Store..." in col. 4 lines 42-43. The classes are common to the objects.)
- grouping together said objects into type sets, wherein each of said objects in each of said type sets have the same type, wherein each of said type sets have a set population equal to a total number of objects inhabiting said type set ("Objects can be stored individually or grouped with other objects" in col. 2 lines 16-17. Further, the sets would inherently have a set population equal to the total number of objects inhabiting the set, as the set population is a direct reflection of what is contained in the set; they would be identical.)
- counting each of said type sets and arriving at a total number of sets ("The class count field..." in col. 6 line 45)
- converting each of said objects to a persistable form including obtaining a persistable form for each of said pointers to objects by obtaining a unique object identifier corresponding to each of said pointers to objects (fill the DataCursor's buffer with class member data, class identifiers and class data offsets..." in col. 5 lines 58-59)
- writing said total number of type sets to persistent storage (Note Figure 8 and the corresponding sections of the disclosure.)
- writing each of said type sets to persistent storage (Note Figure 8 and the corresponding sections of the disclosure.)

substantially as claimed.

Per claim 5:

Greef discloses:

- a method for storing and restoring user objects to persistent storage ("Figure 7 is a flowchart demonstrating 'storing' objects on a flexible nonvolatile or persistence memory" in col. 5 lines 49-50)
- providing one or more common interfaces that are used by the user objects to store and restore the objects to/from persistent storage ("The two primary classes for streaming objects into and out of persistence is the Data Base Cursor and the Data Store..." in col. 4 lines 42-43. The classes are common to the objects.)
- creating a persistence controller object for managing the persistence of the user objects, the persistence controller being derived from at least one of the common interfaces (Note Figure 2 and the corresponding sections of the disclosure)
- providing a plurality of user defined classes, the classes derived from a common object base class (Note Figure 2 and the corresponding sections of the disclosure)
- creating a plurality of instances of user objects belonging to the user defined classes (Each of these classes must also be able to create a new instance of its type..." in col. 5 lines 33-34)
- providing a stream-in method and a stream-out method for each of the user defined classes ("streaming objects into and out of persistence is the Data Base Cursor and the Data Store..." in col. 4 lines 42-43)
- registering each added user defined class and added user object in a registry (Note Figure 2 and the corresponding sections of the disclosure, wherein the system is using a database. Further, "the data base will only save dirty entities..." in col. 4 lines 47-48. The database serves as a registry.)

- grouping the objects according to class (Note Figure 4 and the corresponding sections of the disclosure)
- Storing the grouped user objects to persistent storage using the stream-out methods (Note Figure 7, item 101. The DataStore directs the data cursor to storage the objects.)
- Loading the stored objects from storage into memory using the stream-in methods (Note Figure 9, item 301. The DataStore directs the data cursor to fetch the objects.)
- Registering the user objects in the registry ("the data base will only save dirty entities..." in col. 4 lines 47-48. The database serves as a registry.)

substantially as claimed.

Per claim 6:

The rejection of claim 5 is incorporated, and further, Greef discloses wherein user defined classes have pointers pointing at objects derived from the base class, wherein, when the pointers have a unique identifier associated with the pointer, wherein the saving step includes saving the unique identifiers associated with the pointers, and wherein the loading step includes setting the unique identifier value for each loaded object, obtaining the new address of each loaded user object, and using the stored unique identifier associated with each pointer along with the new address to set each pointer value to the new address as claimed ("This is done by invoking an operation on the Data Store that can find an object in the nonvolatile memory when it is provided with the object's unique identifier. A smart pointer is then created and the smart pointer asks the Class that corresponds to the object's class identifier, to create an object..." in col. 4 lines 55-61)

Per claim 7:

Greef discloses:

- an object adapted for persistent storage, the object having a smart pointer, wherein the smart pointer includes an address attribute for containing the address of an object, and an object unique identifier attribute for containing the unique identifier of an object (Note Figure 1, the Entity Smart Pointer element. Further, "When an operation is invoked on a smart pointer, the object that is points to is faulted..." in col. 4 lines 52-54. The smart pointer inherently contains an address of the object that it points to.)
- wherein the object smart pointer has an assignment operation which stores the address of the object being pointed to and the unique identifier of the object being pointed to ("A smart pointer is then created and the smart pointer asks the Class that corresponds to the object's class identifier..." in col. 4 lines 58-60)
- wherein the object includes a load method for using the smart pointer unique identifier attribute to determine and load a new smart pointer address attribute after the object being pointed to is loaded from persistent storage (Note Figure 10 and the corresponding sections of the disclosure)

substantially as claimed.

Per claim 8:

The rejection of claim 7 is incorporated, and further, Greef discloses a stream-out method for streaming out the smart pointer address attribute and unique identifier attribute as claimed ("All objects manipulate smart pointers. Objects have lists of smart pointers..." in col. 4 lines 39-40. Further, "streaming objects into...persistence..." in col. 4 lines 42-43. Since all objects have smart

pointers, and the system of Greef is streaming objects into persistent storage, then the smart pointer attributes must be streamed to persistent storage as well.)

Per claim 10:

Greef discloses:

- a method for writing computer objects to persistent storage ("Figure 7 is a flowchart demonstrating 'storing' objects on a flexible nonvolatile or persistence memory" in col. 5 lines 49-50)
- providing a plurality of software objects to be stored, wherein the objects are instantiations of at least one class to be storable, wherein the storage is in persistent storage (Note Figure 4 and the corresponding sections of the disclosure)
- wherein each of the classes has a unique class ID, and wherein each of the objects has a unique object ID ("a unique object identifier and a class identifier..." in col. 4 lines 21-22)
- providing smart pointers for at least some objects, wherein the smart pointers include an address portion to contain the address of the object being pointed to and an object identifier portion to contain an objection identifier of the object being pointed to (Note Figure 1, the Entity Smart Pointer element. Further, "When an operation is invoked on a smart pointer, the object that is points to is faulted..." in col. 4 lines 52-54. The smart pointer inherently contains an address of the object that it points to.)
- providing a persistent object controller for controlling the lifecycle of objects to be saved to persistent storage and loaded from persistent storage ("for streaming objects into and out of persistence is the Data Base Cursor and the Data Store..." in col. 4 lines 41-42)

- providing a Persistent Object Registry for maintaining a database of objects to be saved to persistent storage, wherein the Persistent Object Registry is in communication with the persistent controller object (Note Figure 2 and the corresponding sections of the disclosure, wherein the system is using a database. Further, "the data base will only save dirty entities..." in col. 4 lines 47-48. The database serves as a registry.)
- providing a first save method to save all objects in the Persistent Object Registry to persistent storage (Note Figure 7, item 101 and the corresponding section of the disclosure)
- providing a second save method for saving the attributes of each class having objects to be saved to persistent storage, wherein the second save method is called by the first save method (Note Figure 7, item 102 and the corresponding section of the disclosure)
- providing a first load method for loading all objects saved in a file in persistent storage (Note Figure 9, item 301 and the corresponding section of the disclosure)
- providing a second load method for loading the attributes of each class having objects to be loaded from persistent storage, wherein the second load method is called by the first load method (Note Figure 9, item 302 and the corresponding section of the disclosure)
- registering the objects to be saved with the Persistent Object Registry using the persistent object controller, including storing the class ID and object ID of the objects to be saved ("saves it to permanent storage using the object identifier as the unique key" in col. 5 lines 62-63. Further, the data is stored in the database, which acts as the registry.)
- writing the objects to be saved to persistent storage using the first save method and second save method (Note Figure 7 and the corresponding sections of the disclosure)
- reading the objects stored from persistent storage using the first load method and the second load method (Note figure 9 and the corresponding sections of the disclosure)

- resolving the smart pointer object address attributes by using the object ID attribute value to search the Persistent Object Registry to retrieve the current address of the object being pointed to ("invoking an operation on the Data Store that can find an object in the nonvolatile memory when it is provided with the object's unique identifier. The Data Store returns a Data Cursor with the objects contents. A smart pointer is then created and the smart pointer asks the Class that corresponds to the objects class identifier, to create an object of that class" in col. 4 lines 55-61)

substantially as claimed.

Per claim 13:

Greef discloses:

- a method for writing a plurality of objects in non-persistent storage to persistent storage ("Figure 7 is a flowchart demonstrating 'storing' objects on a flexible nonvolatile or persistence memory" in col. 5 lines 49-50)
- providing one or more common interfaces that are used by each of the plurality of objects to write the objects from non-persistent storage to persistent storage ("The two primary classes for streaming objects into and out of persistence is the Data Base Cursor and the Data Store..." in col. 4 lines 42-43. The classes are common to the objects.)
- each of the common object interfaces having a corresponding common object class (Note Figure 2 and the corresponding sections of the disclosure)
- providing a Persistent Object Registry for maintaining a database of objects to be saved to persistent storage, wherein the Persistent Object Registry is in communication with the persistent controller object, and the persistent controller object is derived from one or more

of the common object classes (Note Figure 2 and the corresponding sections of the disclosure, wherein the system is using a database. Further, "the data base will only save dirty entities..." in col. 4 lines 47-48. The database serves as a registry.)

- providing a first save method to save all objects in the Persistent Object Registry to persistent storage (Note Figure 7, item 101 and the corresponding section of the disclosure)

substantially as claimed.

Per claim 14:

The rejection of claim 13 is incorporated, and further, Greef discloses the plurality of objects being derived from one or more common object classes as claimed (Note Figure 5 and the corresponding sections of the disclosure)

Per claim 15:

The rejection of claim 13 is incorporated, and further, Greef discloses a unique object ID as claimed ("unique object identifier..." in col. 4 lines 21-22)

Per claim 16:

The rejection of claim 15 is incorporated, and further, Greef discloses objects having pointers to other objects, and identifying and recording the unique object ID of the other objects as claimed ("Persistent-Object-A has a reference to an instance of Persistent-Object-B..." in col. 5 lines 13-14)

Per claim 17:

The rejection of claim 16 is incorporated, and further, Greef discloses saving the recorded unique object Ids of the other objects as claimed (Note the rejection of claim 16. For the relationship to be preserved, the reference to Object-B from Object-A must be recorded.)

Per claim 19:

Greef discloses:

- a method for reading a plurality of objects in non-persistent storage to persistent storage (Note Figure 9 and the corresponding sections of the disclosure.)
- providing one or more common interfaces that are used by each of the plurality of objects to load the objects from persistent storage to non-persistent storage ("The two primary classes for streaming objects into and out of persistence is the Data Base Cursor and the Data Store..." in col. 4 lines 42-43. The classes are common to the objects.)
- each of the common object interfaces having a corresponding common object class (Note Figure 2 and the corresponding sections of the disclosure)
- providing a Persistent Object Registry for maintaining a database of objects to be saved to persistent storage, wherein the Persistent Object Registry is in communication with the persistent controller object, and the persistent controller object is derived from one or more of the common object classes (Note Figure 2 and the corresponding sections of the disclosure, wherein the system is using a database. Further, "the data base will only save dirty entities..." in col. 4 lines 47-48. The database serves as a registry.)
- providing a load method to save all objects in the Persistent Object Registry to persistent storage (Note Figure 9 and the corresponding section of the disclosure)

substantially as claimed.

Per claims 20 and 21:

The rejection of claim 19 is incorporated, and further, note the rejection regarding claims 14 and 15, respectively.

Per claim 22:

The rejection of claim 21 is incorporated, and further, note the rejection regarding claim 16.

Per claim 23:

The rejection of claim 22 is incorporated, and further, note the rejection regarding claim 17. Similarly, the relationship of the objects would have to be preserved, so the pointed to object ID's would get loaded upon loading Object-A or Object-B.)

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claim 11 is rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 6,125,364 to Greef et al, hereafter referred to as Greef, in view of prior art of record, "Dr. GUI on Components, COM, and ATL", hereafter referred to as Dr. GUI.

Per claim 11:

The rejection of claim 10 is incorporated, and further, Greef does not explicitly disclose objects being Component Object Model (COM) objects. Dr. GUI discloses that the use of COM objects were well known in the art at the time the invention was made ("The Component Object Model (COM) is a way for software components to communicate with each other" on pages 5-6, section titled Ok, so what is COM?). It would have been obvious to one of ordinary skill in the art at the time the invention was made to use COM objects as disclosed by Dr. GUI with the persistent object storage system of Greef, as this would allow a user to retain object information through the persistent storage, while enabling the added benefit of easy component communication regardless of what machine they're running on, as disclosed on page 6 of Dr. GUI.

9. Claim 12 is rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 6,125,364 to Greef et al, hereafter referred to as Greef, in view of U.S. Publication 2003/0163596 A1 to Halter et al, hereafter referred to as Halter.

Per claim 12:

Greef discloses:

- a method for managing persistent object lifecycles ("object persistence framework..." in col. 3 lines 41-42)
- providing for each object a unique object identifier attribute, an object type attribute, and an object address ("a unique object identifier and a class identifier..." in col. 4 lines 21-22. The object inherently has an address in the system.)

- providing an object registry object for maintaining a correspondence between said unique object identifier attribute, said object address, and said object type attributes ("the Data Store that can find an object...when it is provided with the objects unique identifier" in col. 4 lines 55-57. The Data Store is an object of the database registry.)
- creating a first object having a first object type, a first object address, and a first unique object identifier, and storing said first unique object identifier, address, and type in said object registry ("has the object...fill the DataCursor's buffer with class member data, class identifiers and class data offsets...the data store saves it to permanent storage using the object identifier as the unique key..." in col. 5 lines 57-63)
- creating a second object having a second object type, a second object address, and a second unique object identifier, and storing said second unique object identifier, address, and type in said object registry, said second object having a pointer attribute set equal to said first object address (Note Figure 4 and the corresponding section of the disclosure. Persistent-Object-C contains a pointer address to Persistent-Object-B.)
- providing said second object pointer attribute to said object registry and obtaining said first object unique identifier corresponding to said second object pointer attribute in return ("A smart pointer is then created and the smart pointer asks the Class that corresponds to the object's class identifier..." in col. 4 lines 58-60)
- writing said second object to persistent storage as second object data, and writing said first object unique identifier corresponding to said second object pointer attribute to persistent storage, such that said written first object unique identifier is associated with said second object pointer attribute in persistent storage ("If the object is not the top of the class

hierarchy...then the storage method on its super classes are invoked as noted by the query..." in col. 6 lines 2-6)

- reading said second object data from persistent storage and creating said second object having said second object type (Note Figure 10 and the corresponding sections of the disclosure)
- reading said first object unique identifier associated with said second object data from persistent storage ("A smart pointer is then created and the smart pointer asks the Class that corresponds to the object's class identifier..." in col. 4 lines 58-60)
- providing said object registry with said first object unique identifier and obtaining said first object address in return ("the Data Store that can find an object...when it is provided with the objects unique identifier" in col. 4 lines 55-57. The Data Store is an object of the database registry.)
- setting said second object pointer attribute equal to said first object address, such that said second object pointer attribute again points to said first object ("A smart pointer is then created and the smart pointer asks the Class that corresponds to the object's class identifier..." in col. 4 lines 58-60. The smart pointer pointers to the corresponding class. Furthermore, note Figure 4 and the corresponding sections of the disclosure.)

substantially as claimed. Greef does not explicitly disclose deleting said second object from non-persistent storage. Halter discloses in an analogous object-oriented persistence storage system the ability to delete objects from non-persistent storage ("when a reference object is no longer pointed to, the reference object, like any other object can be garbage collected" in paragraph 0070). It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the garbage collecting techniques of Halter with the persistent storage system of Greef, as this would

conserve system resources by removing objects which are no longer needed, as stated in paragraph 0070 of Halter.

10. Claims 18 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 6,125,364 to Greef et al, hereafter referred to as Greef, in view of prior art of record, "Dr. GUI on Components, COM, and ATL", hereafter referred to as Dr. GUI, further in view of U.S. Patent 5,682,536 to Atkinson et al., hereafter referred to as Atkinson.

Per claim 18:

The rejection of claim 13 is incorporated, and further, while Dr. GUI discloses the use of the IUnknown interface, neither Greef nor Dr. GUI disclose an IPersistFile or IPersistStream interface. Atkinson discloses in an analogous system for persistently storing objects that the use of the IPersistFile and IPersistStream interfaces were well known in the art at the time the invention was made. (Note col. 66 line 35 to col. 71 line 13) It would have been obvious to one of ordinary skill in the art at the time the invention was made to utilize the IPersistFile and IPersistStream interfaces so that one could store and restore objects in the system persistently.

Per claim 24:

The rejection of claim 19 is incorporated, and further, note the rejection regarding claim 18.

Response to Arguments

11. Applicant's arguments filed 16 November 2004 have been fully considered but they are not persuasive.

Per claim 1:

The Applicant states that Greef does not teach or suggest objecting a unique object identifier that corresponds to the object pointer for each object pointer in the objects, and obtaining the object address corresponding to the unique object identifier and setting each of the object pointers to the corresponding object address. In response, it is noted that Figure 4 shows objects that are to be stored persistently. Furthermore, "Persistent-Object-A has a reference to an instance of Persistent-Object-B." in col. 5 lines 13-14. As these objects are to be persistently stored, and ultimately restored, then the relationship between the objects must be preserved through storing/restoring of the objects. As such, when Greef creates the object from persistent storage, the pointers and reference of that object must be created as well; otherwise, the object would not be usable as external references would not be valid. The rejection of claim 1 is proper and maintained.

Per claims 2 and 3:

The Applicant states that Greef does not disclose creating objects in a first pass and that the objects' pointers values are set in a second pass subsequent to the first pass. In response, as noted above in regards to claim 1, the relationship structure of objects in Greef must be preserved for the objects to be operational. As such, If Persistent-Object-A has a reference to Persistent-Object-B, then upon restoration of Persistent-Object-A, the reference to Persistent-Object-B would subsequently be set after the recreation of Persistent-Object-A. The rejection of claims 2 and 3 is proper and maintained.

Per claim 4:

The Applicant states that Greef does not disclose providing one or more common interfaces that are used by each of the plurality of objects to write the objects from non-persistent storage to persistent storage. In response, it is noted that the Data Base Cursor and the Data Store are commonly utilized classes for streaming objects into and out of persistence, and as such, qualify as common interfaces used by each object according to the broadest reasonable interpretation of the claim language. The rejection of claim 4 is proper and maintained.

Per claims 5 and 6:

The Applicant states that Greef does not disclose providing one or more common interfaces similarly to claim 4 above. Note the response to the arguments with regard to claim 4. Further, the Applicant states that Greef does not disclose obtaining a new address of each loaded user object and setting each pointer value to the new address. Note the response to the arguments with regard to claim 1. For the relationship between objects to be preserved, the reference, and accordingly the address, of the pointed to object must be set in the restored object. The rejection of claims 5 and 6 is proper and maintained.

Per claims 7 and 8:

The Applicant states that the smart pointers of Greef do not include both an address attribute for containing the address of an object and an object unique identifier attribute for containing the unique identifier of an object, but rather that the smart pointers of Greef appear to only include a unique identifier. In response, it is noted that the smart pointers of Greef "overloads the pointer..." in col. 4 lines 38-39. As such, the smart pointers are still variants of pointers in general, of which operate by reference to memory addresses of objects in the system. Consequently, the smart

pointers of Greef would also contain address attributes of objects in the system, while further providing the new functionality of the overloaded pointer. The Applicant further states that it is the Data Store in Greef, and not the smart pointer, that finds objects in non-volatile memory. However, it is noted that the claim language only requires "*using* (emphasis added) the smart pointer unique identifier..." and, as noted by the Applicant, "the smart pointer appears to store and deliver an object's unique identifier to the Data Store..." (page 21). As such, the Data Store is using the unique identifier of the smart pointer. The rejection of claims 7 and 8 is proper and maintained.

Per claim 10:

The Applicant states that Greef does not disclose or suggest smart pointers including an address portion to contain the address of the object being pointed and an object identifier portion to contain an object identifier of the object being pointed to. Note the response to claim 7 above. Further, the Applicant states that Greef does not disclose or suggest using an object ID attribute value to search the Persistent Object Registry. In response, it is noted that Greef discloses finding objects in persistent storage via an object's ID ("that can find an object in the nonvolatile memory when it is provided with the object's unique identifier..." in col. 4 lines 56-57). Furthermore, Greef discloses the ability to save and load objects from persistent storage, and as noted in the rejection regarding claim 10, discloses the required limitations of saving and loading the objects. For the objects to be persistently stored and restored, their attributes must be stored and restored as well. The rejection of claim 10 is proper and maintained.

Per claim 12:

The Applicant states that Greef does not disclose setting said second object pointer attribute equal to said first object address, such that said second object pointer attribute again points to said first object. In response, note the response regarding claim 1 above. As the relationship between Persistent-Object-A and Persistent-Object-B is maintained during storing and restoring procedures, then the reference pointer must be set so that the second object pointer points to the first object. The rejection of claim 12 is proper and maintained.

Conclusion

12. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Trent J. Roche whose telephone number is (571) 272-3733. The examiner can normally be reached on Monday - Friday, 9:00 am - 5:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571) 272-3719. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Trent J Roche
Examiner
Art Unit 2193

TJR

ANIL KHATRI
PRIMARY EXAMINER